APLICAR AUTENTICAÇÃO
+
CONHECER AUTORIZAÇÃO

REPROGRRAMA - SEMANA 14 -
BACKEND - PROF MAYHHARA

OIII, GATINHASSSS

Eu sou a May!
Sou líder técnica da turma On19, engenheira de software backend e, atualmente, trabalho (principalmente) com GoLang na Wildlife Studios.

SAC da May

Mayhhara Morais:	linkedinho
Mayhhara Morais:	slack
@mayhhara_:	instagram
@mayjinboo:	twitter

AGENDA DE HOJE

DICA

Afim de mantermos a fluidez da aula, deixarei disponível um link para o Slido, onde vocês conseguirão adicionar as dúvidas que surgirem durante as explicações, assim elas nao ficarão perdidas no chat.

Sábado, 12/Nov

- Segurança
- Criptografia
- Autenticação Vs Autorização
- Hash
- Criptografia Simétrica Vs C. Assimétrica
- OAuth
- Jwt
- Mão na massa!

FRASE DO DIA

Voce conhece a Gi?
A Gigantesca capacidade que você tem de superar seus problemas! Levanta essa cabeça, mate o desanimo na paulada, destrua as adversidades na bicuda! Lembre-se: você é uma máquina de vencer!

Clarice Lispector

SEGURANÇA

Nas aulas anteriores você aprendeu sobre rotas **POST**, **GET**, **PATCH**, **PUT** e **DELETE**. Aprendeu também sobre **banco de dados** e também a utilizar essas rotas para trabalhar com os dados. Entretanto, concorda que qualquer pessoa hoje que tiver acesso a essas rotas que criou poderá utilizá-las livremente para salvar, trazer, alterar e deletar informações sem a necessidade de controle?

Como você acha que podemos aumentar a segurança das rotas da nossa API?

AUTENTICAÇÃO VS AUTORIZAÇÃO

AUTENTICAÇÃO

É a checagem da identidade de um usuário ou sistema. Existem várias formas de autenticação, como por exemplo login com usuário e senha, autenticação biométrica etc.

AUTORIZAÇÃO

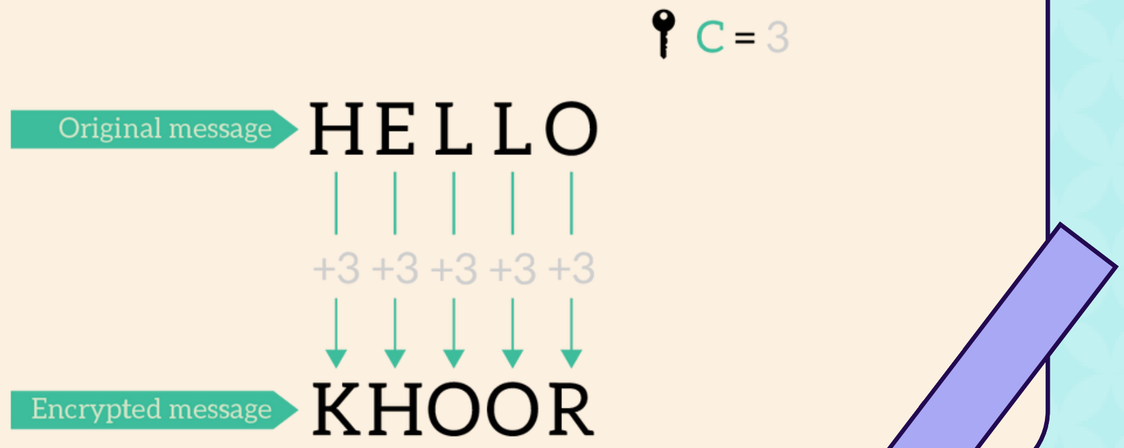
É a checagem de permissão de um usuário autenticado.

Como você armazenaria a senha desse usuário na base de dados para uma autenticação?

PLUS:
Um filme bem bacana que fala desse tema e que vale a pena assistir é o Jogo da Imitação, que conta sobre o matemático Alan Turing e a criptoanalista Joan Clarke na Segunda Guerra Mundial, quando desenvolveram uma máquina que foi capaz de descriptografar (revelar) mensagens de comunicação da Alemanha Nazista.

CRIPTOGRAFIA

A criptografia é um conjunto de técnicas pensadas para proteger uma informação de modo que apenas o emissor e receptor consigam compreendê-la.
Em outras palavras: é **codificar uma mensagem que você não quer que pessoas não autorizadas tenha acesso.**



C. SIMÉTRICA

A criptografia simétrica é o tipo mais tradicional e provavelmente o sistema que as pessoas estão mais familiarizadas.
Nele, a criptografia é realizada com base em uma única chave — que é utilizada para criptografar e também descriptografar uma mensagem.
O exemplo da imagem anterior ilustra bem a criptografia simétrica:
A chave é 3 e a mensagem original, “**HELLO**“, foi criptografada como “**KHOOR**“.
Ou seja, para descriptografar, basta aplicar a mesma chave/lógica, que é a troca de cada letra para a terceira letra posterior a ela no alfabeto.

Sua principal aplicação é na proteção de dados em repouso, como em bancos de dados ou discos rígidos — isso porque é necessário contar com um canal seguro para transmitir a mensagem.

Os algoritmos mais conhecidos são: DES, TripleDES, AES, RC4 e RC5.

C. ASSIMÉTRICA

A criptografia assimétrica utiliza duas chaves diferentes para criptografia e descriptografia de um dado.
Vamos direto ao ponto: a primeira chave é uma chave pública usada para criptografar uma mensagem e a segunda é uma chave privada utilizada para descriptografá-la.
O principal a se entender é que apenas uma chave privada pode descriptografar as mensagens criptografadas por uma chave pública.

A criptografia assimétrica é aplicada em várias operações do dia a dia, **como assinatura eletrônica, envio de e-mails ou mesmo realizar uma conexão remota a um sistema privado.**
O principal **benefício** da criptografia assimétrica é que as **pessoas não precisam de nenhum esquema de segurança específico para trocar mensagens com confidencialidade.**

Os algoritmos mais conhecidos são: RSA e ECDSA.

SIMÉTRICA X ASSIMÉTRICA

01

SIMÉTRICA

- Chaves mais simples, fácil processamento, exige pouco poder computacional.
- A chave tem que ser compartilhada entre os envolvidos, o que pode se tornar uma vulnerabilidade.
- Pode ser usada nos dois sentidos (A -> B) e (B <- A), já que os dois usuários usam a mesma chave.

02

ASSIMÉTRICA

- Chaves maiores, que exigem hardware mais poderoso e mais tempo para processamento.
- Apenas a chave de codificação é compartilhada. A chave decodificada continua segura!
- Só pode ser usada em um sentido (A -> B), usando as chaves de B ou no outro sentido (B -> A), usando as chaves de A.

Para armazenar senhas na base de dados, por exemplo, podemos utilizar um hash. Mas o que seria hash?

HASH

O algoritmo Hash é conhecido como uma função matemática criptográfica, na qual você possui dados de entrada e, após passar pela criptografia, eles apresentam valores de saída "padronizados", ou seja, as saídas devem possuir o mesmo tamanho (geralmente entre 128 e 512 bits) e o mesmo número de caracteres alfanuméricos.

RESUMINDO: A função hash criptográfica é utilizada, principalmente, para resumir uma grande quantidade de informações em arquivos.

Imagine um banco de dados com muitas informações podendo ser resumido em uma única sequência de letras e números! Bem mais prático, né?!

A entrada pode ser um arquivo de mídia, senhas, banco de dados, docs, entre outros

A função hash criptografa os dados de entrada

28a905fb82ec64b0a75badc8bf029c660856781a

431fede2c383be77cf8e948602e27e30bdfe1f43

24a2be97ff3177ad22486d53d9a97f23fd819848

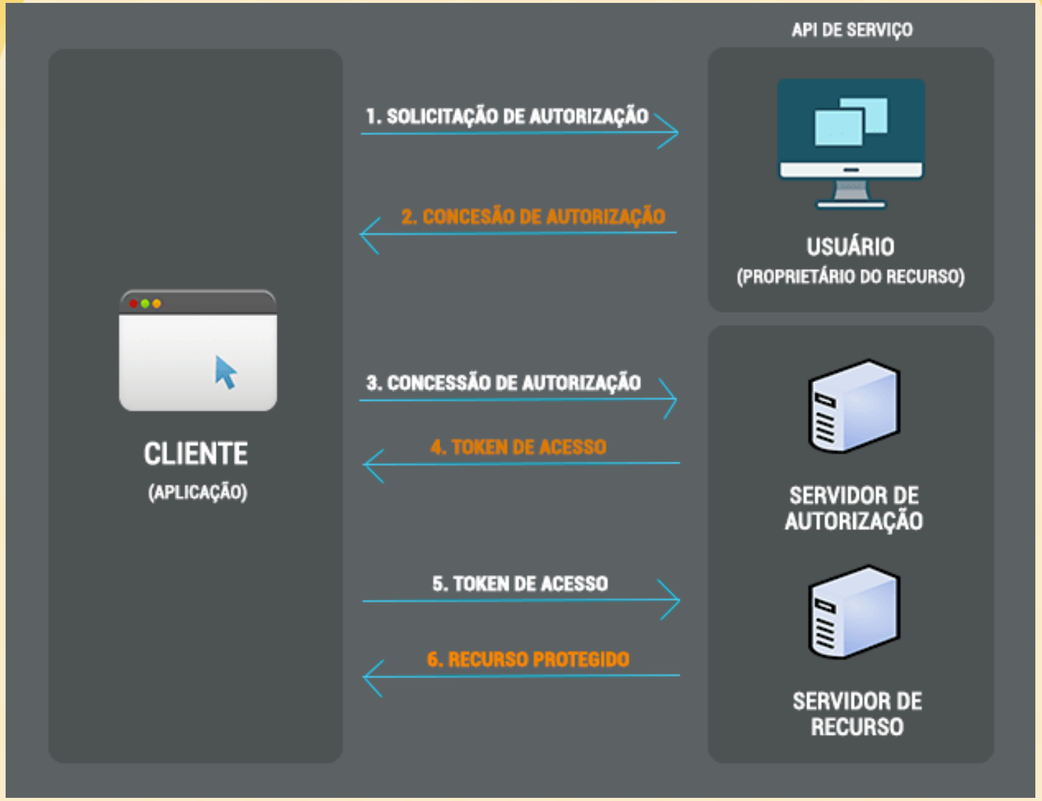
Saídas alfanuméricas padronizadas

Importante: Diferente do tipo simétrico e assimétrico, uma função de hash foi projetada para ser irreversível. Uma vez que você transforma uma string em um hash, a partir de uma função de hash, não é possível transformar novamente na string original.

Algoritmos mais conhecidos para hash: MD5, SHA-1 e SHA-2.
(É possível criptografar usando esses algotimos pelo terminal (:)

Vantagem: É uma operação pouco custosa de computação e segura pois é unidirecional, isto é, impossível de você voltar a string original a partir do hash.

Desvantagem: a principal desvantagem é que não é possível recuperar uma senha; você só pode redefinir sua senha.



1. **Solicitação de autorização:** nessa primeira etapa o cliente (aplicação) solicita a autorização para acessar os recursos do servidor do usuário
2. **Concessão de autorização:** se o usuário autorizar a solicitação, a aplicação recebe uma concessão de autorização.
3. **Concessão de autorização:** o cliente solicita um token de acesso ao servidor de autorização (API) através da autenticação da própria identidade e da concessão de autorização.
4. **Token de acesso:** se a identidade da aplicação está autenticada e a concessão de autorização for válida, o servidor de autorização (API) emite um token de acesso para a aplicação. O cliente já vai ter um token de acesso para gerenciar e a autorização nessa etapa já está completa.
5. **Token de acesso:** quando o cliente precisar solicitar um recurso ao servidor de recursos, basta apresentar o token de acesso.
6. **Recurso protegido:** o servidor de recursos fornece o recurso para o cliente, caso o token de acesso dele for válido.

OAuth

Provavelmente você já clicou em algum botão escrito “Logar com sua conta do Google” ou “Logar com sua conta do Facebook” quando você está em algum site , para evitar ter que realizar um novo cadastro com senha e afins em um serviço.

Neste caso, você está dando a autorização de uma aplicação de terceiros a usar os recursos de suas contas no Google ou o Facebook como credenciais. Isso constitui o protocolo OAuth .

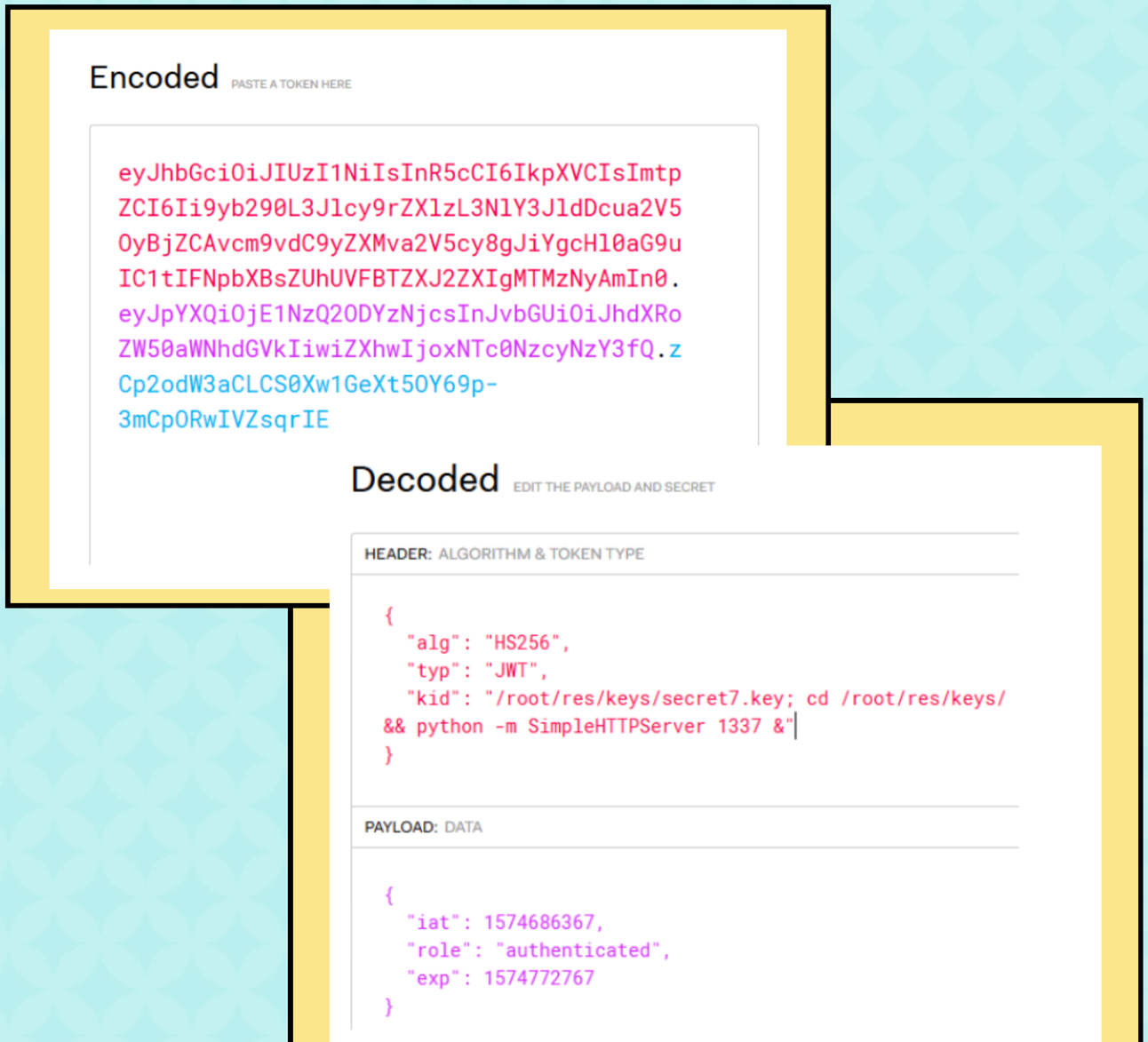
OAuth é um protocolo de autorização que permite que uma aplicação se autentique em outra, como visto no exemplo acima.

JSON WEB TOKEN

JWT - CONCEITO

O padrão JWT permite que as informações sejam assinadas tanto com criptografia simétrica (com o algoritmo HMAC) quanto com criptografia assimétrica (com os algoritmos RSA e ECDSA).

Os JWTs são muito utilizados no processo de autenticação permitindo que o processo de autorização de acesso a recursos seja mais rápido e escalável. Mais rápido porque por ser independente retira da equação o tempo de latência (tempo que demora para iniciar e terminar) de acesso ao banco de dados ou outro mecanismo de cache. E mais escalável pois permite que serviços totalmente independentes compartilhem a mesma autenticação sem necessitar de comunicação entre os mesmos.

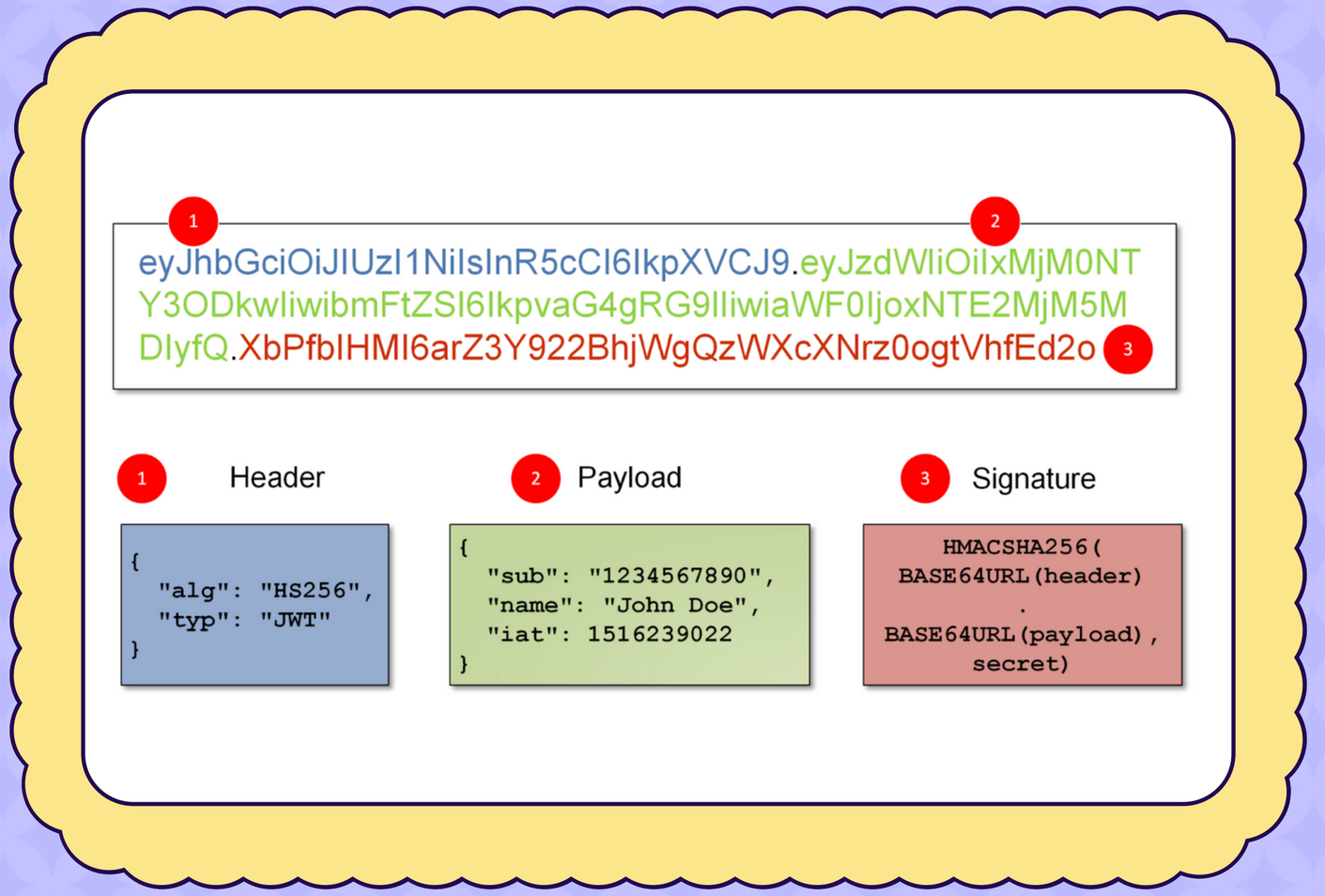


JWT - ESTRUTURA

Header: É um objeto JSON que define informações sobre o tipo do token (typ), nesse caso JWT, e o algorítmo de criptografia em sua assinatura (alg), normalmente HMAC SHA256 ou RSA.

Payload: É um objeto JSON com as Claims (informações) da identidade tratada, normalmente o usuário autenticado.

- **Reserved claims:** atributos não obrigatórios que são usados na validação do token pelos protocolos de segurança das APIs. Os mais utilizados são: **sub** (entidade a quem o token pertence), **iss** (emissor do token) e **exp** (timestamp de quando o token irá expirar);
- **Public claims:** atributos que usamos em nossas aplicações. Normalmente armazenamos as informações do user autenticado na aplicação como nome, roles, permissões, por exemplo.
- **Private claims:** atributos definidos especialmente para compartilhar informações entre aplicações. (Por segurança, recomenda-se não armazenar infos confidenciais ou sensíveis no token).



JWT - ESTRUTURA

Signature: A assinatura é a concatenação dos hashes gerados a partir do Header e Payload usando base64UrlEncode, com uma chave secreta ou certificado RSA.

Essa assinatura é utilizada para garantir a integridade do token, no caso, se ele foi modificado e se realmente foi gerado por você.

RESULTADO FINAL

O **resultado final** é um token com três seções (**header, payload, signature**) separadas por **“.” — ponto**.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJz
dWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gR
G9lIiwiYWRtaW4iOnRydWV9.TJVA95OrM7E2cBab3
0RMHrHDcEfxjoYZgeFONFh7HgQ
```

JWT - VULNERABILIDADES

- Se a biblioteca aceita que um token seja validado sem especificar o algoritmo esperado, outra vulnerabilidade grave é aberta.
- Exatamente no caso esperarmos que o token use uma criptografia assimétrica e o atacante utiliza uma criptografia simétrica.

- O problema com essa lógica é que o atacante pode obter a chave pública e assinar um token qualquer utilizando um algoritmo simétrico (HMAC) e indicar no cabeçalho o mesmo algoritmo.
- Assim quando um recurso protegido utilizar o mesmo algoritmo e a mesma chave o token será considerado válido, pois a assinatura gerada será igual a assinatura do token.

- Lembrando que nesse caso como os tokens válidos estão sendo assinados com a chave privada os mesmos devem ser validados com a chave pública. Por isso o atacante terá sucesso, pois tem a certeza que o token está sendo validado com a chave pública.

AUTENTICAÇÃO NA PRÁTICA

FLUXO AUTENTICAÇÃO

1) Criação de usuária
Uma usuária é criada e sua senha é armazenada como um hash (usando o bcrypt).

2) Login da usuária
Na request de login, são enviados no body os dados necessários para autenticação (email e senha, por exemplo).

3) Autenticação da usuária
A senha é verificada com a do banco, se for igual, um código(token) é gerado como resposta à requisição. No front, esse código(token) é armazenado.

4) Autorização de visualização
Com o login realizado, a cada nova requisição o token é enviado no body da requisição permitindo a autorização de visualização.


HORA DE
COLOCAR A MÃO
NO CÓDIGO!
****

REFERENCIAS

- https://tecnoblog.net/responde/o-que-e-um-hash/
- https://www.voitto.com.br/blog/artigo/o-que-e-hash-e-como-funciona
- https://www.wellingtonjhn.com/posts/entendendo-tokens-jwt/
- https://canaltech.com.br/seguranca/o-que-e-o-protocolo-oauth/
- https://blog.mailfence.com/pt/criptografia-simetrica-x-assimetrica-qual-e-a-diferenca/

VALE DAR UMA LIDA

obrigada,
meninas!